CHAPTER 1

Variables in the Web Design Environment

When you complete this chapter, you will be able to:

- Describe the current state and design limitations of HTML
- Learn how XML and XHTML could transform the future of the Web
- Describe how Web browsers affect the way users view your Web site
- Decide if you should use an HTML editor to create markup code
- Describe how screen resolution and connection speed affect the user's browsing experience
- Clear the cache when testing your site
- Describe how Web pages are delivered via the Internet and stored on a user's computer

In this chapter you will explore the variable factors that affect Web design. You will learn how **Hypertext Markup Language** (**HTML**), the authoring language used to create documents on the World Wide Web, is constantly evolving, and preview the new markup languages that will change how you design for the Web. You will see how Web browsers affect the way users view your content, and whether using an HTML editor can enhance your HTML coding skills. You also will learn how the user's browser choice, screen resolution, and connection speed pose specific challenges to creating Web pages that display properly in different computing platforms.

HTML, XML, AND THE FUTURE OF MARKUP LANGUAGES

In this section you will explore the roots of HTML and the future of markup languages for creating Web documents. You will analyze current design limitations of HTML, the use of Cascading Style Sheets, and the potential uses of XML and XHTML, which are new markup languages expected to affect the Web's future.

HTML: THEN AND NOW

When Tim Berners-Lee first proposed HTML at the European Laboratory for Particle Physics (CERN) in 1989, he was looking for a way to manage and share large amounts of information among colleagues. He proposed a web of documents (at first, he called it a mesh) connected by hypertext links and hosted by computers called hypertext servers. As the idea developed, Berners-Lee named the mesh the World Wide Web. He created an application of the **Standard Generalized Markup Language (SGML)**, a standard system for specifying document structure, and called it the Hypertext Markup Language (HTML). HTML greatly reduces the complexity of SGML to enhance transmission over the Internet.

When Berners-Lee created HTML, he adopted only the necessary elements of SGML for representing basic office documents, such as memos and reports. The first working draft of HTML included elements such as titles, headings, paragraphs, and lists. HTML was intended for simple document structure, and not to handle the variety of information needs in use today. The document expression capabilities of HTML currently are pushed to the limit. HTML has been manipulated in thousands of ways to encompass the staggering breadth of information available on the Web.

Hypertext and Hypermedia

The basic concept of the World Wide Web, and the most revolutionary idea envisioned by Berners-Lee, is the use of hypertext to link information on related topics over the Internet. **Hypertext** is a non-linear way of organizing information. When using a hypertext system, you can jump from one related topic to another, quickly find the information that interests you, and return to your starting point or move onto another related topic of interest. As a hypertext author, you determine which terms to create as hypertext links and where users will end up when they click a link.

On the Web, clickable hyperlinks, which either can be text or images, can connect you to another Web page or allow you to open or download a file, such as a sound, image, movie, or executable file. The early Internet consisted of only text and binary files, and when these new hypertext capabilities were introduced, they demanded a new term—**hypermedia**, the linking of different types of media on the World Wide Web.

HTML as a Markup Language

A **markup language** is a structured language that lets you identify common sections of a document such as headings, paragraphs, and lists. An HTML file includes text and HTML markup elements that identify these sections. The

HTML markup elements indicate how the document sections display in a browser. For example, the <H1> element tags in the following code indicate that the text is a first-level heading:

```
<H1>This is a first-level heading.</H1>
```

The browser interprets the HTML markup elements and displays the results, hiding the actual markup tags from the user. In the above code, the user sees only the text "This is a first-level heading." formatted as a level-one heading.

HTML adopts many features of SGML, including the cross-platform compatibility that allows different computers to download and read the same file from the Web. Because HTML is cross-platform compatible, it does not matter whether you are working on a Windows PC, Macintosh, or UNIX computer. You can create HTML files and view them on any computer platform.

HTML is not a What You See Is What You Get (WYSIWYG) layout tool. It was intended only to express logical document structure, not formatting characteristics. Although many current HTML editors let you work with a graphical interface, the underlying code they create still is basic HTML. Because HTML was not designed as a layout language, however, many editing programs create spaghetti code or use HTML tricks to accomplish a certain effect. You cannot rely on the HTML editor's WYSIWYG view to test your Web pages. Because users can view the same HTML file through different browsers and on different machines, the only way to be sure of what your audience sees is to preview your HTML files through the browsers you anticipate your audience will use.

Despite its limitations, HTML is ideal for the Web because it is an open, non-proprietary, cross-platform compatible language. All of the markup tags are included with every document and usually can be viewed through your browser. Once you are familiar with the HTML syntax, you will find that one of the best ways to learn new coding techniques is to find a Web page you like and view the source code.

Note: To view a page's source code in Netscape Navigator 4.0, click View on the menu bar, and then click Page Source. The page's source code opens in another window of Netscape. You can copy and paste text from this window into your own text or HTML editor, and then manipulate and test the code. You cannot edit directly in the Page Source window. In Internet Explorer 4.0 or 5.0, click View on the menu bar, and then click Source. Notepad opens and displays the page's source code. You then can save the file to your own hard drive and manipulate the code. Whenever you copy code from a Web site, remember to respect the author's copyrights to any original material. Although page layouts cannot be copyrighted, any original text or graphics are the property of the author and should be properly cited.

HTML and the World Wide Web Consortium

HTML has progressed significantly since it was first formalized in 1992. After the initial surge of interest in HTML and the Web, a need arose for a standards organization to set recommended practices to guarantee the open nature of the Web. The **World Wide Web Consortium** (W3C®) was founded in 1994 at the

Massachusetts Institute of Technology to meet this need. The W3C, led by Tim Berners-Lee, sets standards for HTML and provides an open, non-proprietary forum for industry and academic representatives to add to the evolution of this new medium. The unenviable goal of the W3C is to stay ahead of the development curve in a fast-moving industry. Since its founding, the W3C has set standards for a markup language that is being changed by the evolution of browsers from competing companies, each trying to claim its share of Web users.

Browser Chaos

As different browsers tried to attract market share, a set of proprietary HTML elements evolved that centered around the use of each particular browser. Some examples of these elements are <FONT> and <CENTER>, which were developed specifically for the Netscape browser. <FONT> eventually became part of the HTML 3.2 specification, but it has been designated as a deprecated element in HTML 4.0. **Deprecated elements** are those that the W3C has identified as obsolete and will not be included in future releases of HTML. It is likely, however, that these elements and others like them will be supported by browsers for some time. The browser developers would be doing users a disservice (and possibly losing customer share) if they removed support for these elements.

Adding to this confusing compatibility issue are the elements that are strictly proprietary, such as <MARQUEE> (Internet Explorer only), which creates scrolling text, and <BLINK> (Netscape Navigator only), which makes text blink on and off. These elements work only within the browser for which they were designed and are ignored by other browsers. Using proprietary elements like these defeats the open, portable nature of the Web. They are not included in the standard maintained by the W3C. Avoid using proprietary elements unless you are sure that your audience is using only the browser for which they were designed.

An alternative browser is Opera from *www.operasoftware.com*. Developed in Norway, Opera is very popular in Europe. Opera is a fast, lean browser that does not include unnecessary add-ons—a refreshing alternative to the hard drive space demands of Netscape and Internet Explorer. If you are developing a site that will have international exposure (it is the *World Wide Web*, after all), consider adding Opera to your set of test browsers.

TIP

Visit the W3C site at *www.w3.org* to find out more about HTML, XML, CSS, and the history and future of the Web. You can look up individual element definitions, test your code for validity, or keep up to date on the latest Web developments.

Separating Style from Structure

Style elements such as <FONT> were introduced by browser developers to help HTML authors bypass the design limitations of HTML. Designers and writers who are accustomed to working with today's full-featured word processing programs want the same ability to manipulate and position objects precisely on a Web page as they have on the printed page. This is not what HTML was designed to do, however. HTML, like SGML, is intended to represent document structure, not style. Style information should be separate from the structural markup information.

This separation of style and structure has been accomplished by the W3C, which wrote a specification for a Web style language in 1996. The style language, named **Cascading Style Sheets (CSS)**, allows authors to create style rules for elements and express them externally in a document known as a **style sheet**. CSS rules are easy to create and very powerful. For example,

assume that you want all of your <H1> headings to appear green and centered everywhere on your Web site. For every instance of an <H1> element, you would have to include the following code in a standard HTML document:

```
<FONT COLOR="GREEN"><H1 ALIGN="CENTER">Some Heading
Text</H1></FONT>
```

Using a CSS rule, you can express the same style as follows:

```
H1 {COLOR: GREEN; TEXT-ALIGN: CENTER}
```

You can place this rule in an external style sheet, and then link every page on your site to that style sheet. With a minimum of code you can express the same result. Later, if you want to change the <H1> color to red, you simply change the style sheet rule to change every page on your site.

This mixing of structure and style and the subsequent varied support of different style elements in different browsers threaten to undermine the entire foundation of the Web and to render it useless as HTML becomes fractured by proprietary elements. Until recently, the adoption of CSS as a standard for style has been limited because of poor support by the major browsers. Both Internet Explorer and Netscape Navigator plan to offer better support for this powerful style language.

XML: AN OPEN STANDARD FOR STRUCTURING DATA

With the **Extensible Markup Language** (**XML**), the W3C has a chance to start from scratch in defining a standard markup language, instead of playing catch-up as it had to do with HTML. XML is also a subset of SGML. Unlike HTML, XML is a **meta-language**, not a language itself, but a language that lets you describe other languages. As a meta-language, XML allows you to create your own elements to meet your information needs, which significantly distinguishes it from the predefined elements of HTML. XML provides a format for describing structured data that can be shared by multiple applications across multiple platforms.

XML Describes Data

The power of data representation in XML comes from separating display and style from the structure of data. XML elements describe data and structure only and not presentation. Where HTML contains elements that describe a word as bold or italic, XML declares an element to be a book title, item price, or product measurement. Once the data is structured in XML, it can be displayed across a variety of media, such as a computer display, television screen, or handheld device, using an associated style sheet that contains the appropriate display information. Currently, Cascading Style Sheets is the only complete style language for XML, though the W3C is working on the **Extensible Style Language** (**XSL**), which is derived from XML.

XML Allows Better Access to Data

XML is valuable because it allows access to more meaningful searches for information, development of flexible applications, multiple views of data, and is based on a non-proprietary standard that supports the open nature of the Web. Consider the following example.

Suppose you want to sell audio CDs on the Web. Currently, the markup code for a page from your online catalog looks like the following:

```
<BODY>
     <H1>Songs of the World</H1>
     <H2>Artist: World Singers</H2>
     <P>Format: CD</P>
     <P>Price: $12.95</P>
     <P>Track List<P>
     <UL>
     <LI>A Song
     <LI>Another Song
     </UL>
</BODY>
```

Using XML, you could code your catalog this way:

```
<RECORDING>
     <TITLE>Songs of the World</TITLE>
     <ARTIST>Artist: World Singers</ARTIST>
     <FORMAT>CD</FORMAT>
     <PRICE>12.95</PRICE>
     <TRACKLIST>
     <TRACKTITLE NUMBER="1">A Song</TRACKTITLE>
     <TRACKTITLE NUMBER="2">Another Song</TRACKTITLE>
     </TRACKLIST>
</RECORDING>
```

Notice how the data is structured by the elements in the two preceding examples. The HTML elements have no inherent meaning or connection to the data. They simply describe text types and display information. If a customer wants to search for a particular recording, the search program must read each line of code to find the title or artist based on the content. In contrast, the XML code has meaningful element names, which could match the field names in a database. The search program could look for a particular element quickly that matches the user's request. An invoicing program could extract the price easily and list it on the customer's invoice. The possibilities are endless because the element names match and describe the data they contain.

People as well as machines can read XML code. Some years from now, someone could open your XML file and understand what you meant by using <HEADLINE> and <PARAGRAPH> as markup elements. The cross-platform, independent nature of XML markup supports a variety of data applications migrating to the Web. XML lets you display data on many devices without changing the essential data descriptions.

XML Lends Itself to Customized Information

XML is ideal for distributing complex, esoteric information among many users sharing the same types of knowledge. With the customized elements of XML, standards organizations for various industries and information types can enforce

the use of markup constraints with all users. For example, chemists can have their own chemical markup language, and poets can have their own poetic markup language—ensuring that the content can be read and understood by anyone interested in their information type.

Currently, XML still is under development. Many of the proposed features of XML, including XML linking and the Extensible Style Language, still are incomplete and supported differently by different vendors. Internet Explorer 5.0 is currently the only browser that supports XML, and its support is weak in some areas. Be careful with implementation of any XML project by testing your work thoroughly.

XHTML: THE FUTURE OF HTML

HTML has progressed through a number of versions since its inception. The latest standard is version 4.0, which was released by the W3C in late 1997. This probably will be the last version of HTML in its current state. The W3C is working on the next generation, which will be HTML as an application of XML. This new markup language is called the **Extensible Hypertext Markup Language (XHTML)**.

HTML as an Application of XML

The W3C decided that the best way to move forward with HTML is to rebuild it as an application of XML. The W3C released the XHTML 1.0 standard in August of 1999. It replaces and extends the capabilities of HTML 4.0. XHTML should display properly in browsers that support HTML 4.0, but will also be XML compliant. XML is extensible, meaning that authors can create their own elements.

The next release of HTML will include a base element set for most common markup elements such as headings, paragraphs, lists, hypertext links, and images. Other facets of HTML, including forms, tables, frames, and multimedia, will be contained in separate element sets. These add-ons will be expressed using XML syntax and can be combined into your document as needed.

Improved Data Handling

Because XML allows better data handling, the new version of XHTML will work better with database and workflow applications. The next generation of HTML will include advanced support for form elements, defining them more for data handling than presentation, and allowing data to be passed between applications and devices with greater ease. Tables will emphasize a data model that allows their content to be rendered based on the presentation device. For example, tabular data for stock pricing information could be sent to multiple destinations, including a computer monitor, pager, and ticker device. The same data has greater value because it only needs to be generated once but can be displayed in many ways.

Style Sheets Are Required

Because XHTML is an application of XML, you must use style sheets to render style in XHTML. Because data is separate from style, the same information can be directed to various display devices simply by changing the style sheet. By using different style sheets, the contents of the same Web page can be displayed on a computer monitor, TV screen, handheld device, or cellular phone screen.

This data-once, destination-many format liberates the data and structure of XHTML documents to be used in a variety of applications. A script or applet would redesign the data presentation as it is requested from the server and apply the proper style sheet based on the user's choice of device.

HOW WEB BROWSERS AFFECT YOUR WORK

One of the greatest challenges facing HTML authors is designing pages that display properly in multiple browsers. Every browser contains a program called a **parser** that interprets the markup tags in an HTML file and displays the results in the **canvas area** of the browser interface, as illustrated in Figure 1-1. The logic for interpreting the HTML tags varies from browser to browser, resulting in many, possibly conflicting interpretations of the way the HTML file is displayed. As a Web page designer, you must test your work in as many different browsers as possible to ensure that the work you create appears as you designed it. Although you may consider your work cross-browser compatible, you may be surprised to see that the results of your HTML code look very different when viewed with different browsers.

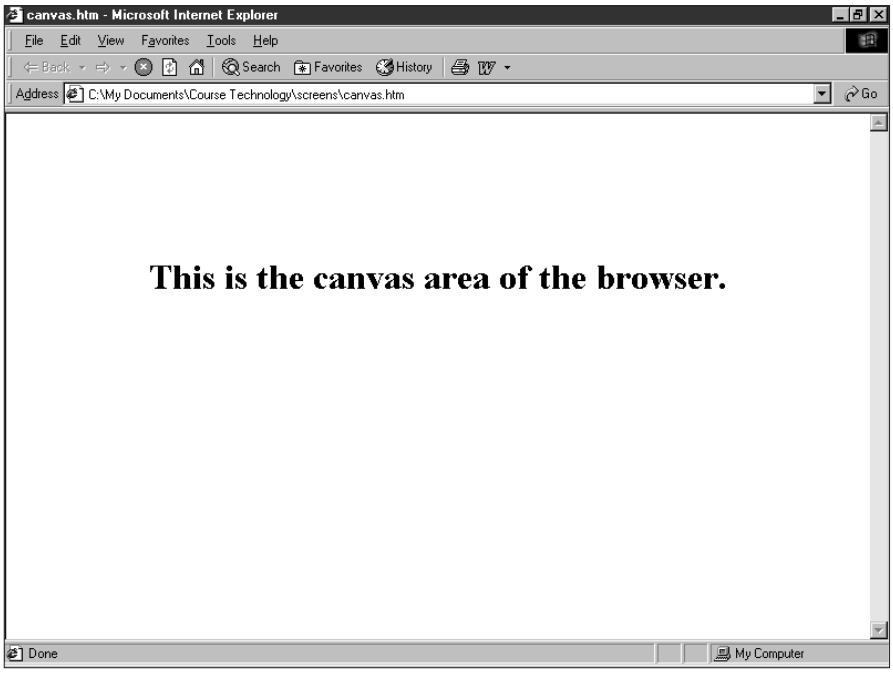


FIGURE 1-1
Canvas area of the browser

Most HTML authors do not have the luxury of knowing which browser each person uses to view their Web pages. Browser and version choices can vary widely based on a number of variables. Do not assume that your users always have the latest browser or operating system. Many individuals and organizations are reluctant to upgrade software simply because a new version has been released. Other users may

have older computers that do not have the processing speed or disk space to handle a newer browser. Although it is a good idea to test with the latest browsers, it also is prudent to test your work in older browsers to ensure that the greatest number of people are seeing your Web pages as you intended your pages to look.

As discussed above, not only are new browsers released frequently, but older browsers still are used by many Web users. The newer browsers support desirable features, such as Cascading Style Sheets, that are not supported by older browsers. Including newly supported features in your page design may significantly affect the way your page is viewed if the browser cannot interpret the latest enhancements. Browsers exhibit subtle differences across computing platforms as well. For example, Netscape for Windows is not exactly the same as Netscape for the Macintosh.

TIP

If you would like to download a particular browser, or find out which browser is currently the most popular, visit one of these Web sites:
Browsers.com at *www.browsers.com*
BrowserWatch at *www.browserwatch.com*
Which Browser at *www.whichbrowser.com*

You can handle the demands of different browsers while designing attractive Web pages. Some HTML authors suggest that you use an older version of HTML to ensure portability. Others say that you should push the medium forward by coding to the latest standard and using the most recent enhancements. Some Web sites recommend that you use a particular brand and version of browser to access the site. Let us examine each of these methods to determine the best way to design your site.

LOWEST COMMON DENOMINATOR CODING

Although it may seem difficult to create pages that always display properly, it is not impossible. One way to create portable pages is to use a lowest common denominator approach. This approach provides the greatest acceptance across browsers because the authors choose to code their HTML using the next-to-last release of HTML. For example, when the browsers supporting HTML 4.0 were released, many continued coding to the HTML 3.2 standard, knowing that their HTML would render more consistently because the browsers understood all of the 3.2 specifications. This safer method of coding is widely supported among sites that are interested in the greatest accessibility. Maintaining coding specifications of a previous release of HTML does not mean that your site has to be visually uninteresting, although you may have to sacrifice using the latest enhancements. Sound visual and information design techniques, discussed in Chapter 2, can let you overcome many functional limitations.

CUTTING-EDGE CODING

Another strategy to adopt when designing your Web site is to stay at the cutting edge. Some designers insist that their users keep up with them by requiring the latest browser. This design strategy can result in visually exciting and interactive sites that keep pace with the latest technology. Often the user must not only have the latest browser version, but plug-in enhancements that render certain media types such as Macromedia Flash animations. **Plug-ins** are helper applications that assist a browser in rendering a special effect. Without the plug-in, your user will not see the results of your work. Often when a new browser is released, these plug-ins are included for the most widely adopted enhancements. The risk these cutting-edge sites take is that many users may not be able

to see the content as it was designed. Sites that use the latest enhancements also may require significant download times for the special effects to load on the user's computer. Sites that adopt the latest technologies must make sure that their user is up to the browser challenge. Otherwise, their information may go unread.

BROWSER-SPECIFIC CODING

How many times have you visited a site that states "This site best viewed with Internet Explorer 4.0"? This statement warns you that the author has decided to forgo the challenge of coding for multiple browsers. The author may have wanted to use some unique enhancement for the site, or may have found that the site did not render properly in other browsers. This may seem the most expedient coding method to adopt. Consider the consequences, however. A site coded for only one browser may alienate a significant number of readers who immediately leave because they do not have the correct browser. On the Web, you never can be sure of the type of browser your user has. This method of browser-specific coding, however, may be viable on a company **intranet**, where you know or you can specify that all users have the same version and brand of browser. For the general Web it is the least desirable choice because you are limiting the availability of your site.

SOLVING THE BROWSER DILEMMA

You must test your work in as many browsers as possible during and at the end of the development process to make sure that your pages will render properly. Knowing your audience helps you take a major step towards the correct implementation of your site. For example, you may be building a site that discusses the latest in technology trends. It is a good bet that your users are computer-savvy, so you can code for the latest browsers. Many general Web users access the Web via America Online, Inc. (AOL), so test your work using their browser as well. In an academic environment you may encounter readers that use Lynx, a text-only browser. For this type of audience, avoid using too many graphics, and make sure that all of the graphics you include have ALT attributes, which provide alternate text information about images. Finally, if you are designing for an audience that has physical challenges, keep their considerations in mind. You can find out more about this special audience at the W3C accessibility initiative page: *www.w3.org/WAI/*.

Note: You can check your Web pages for ease of accessibility to physically-challenged people by using Bobby, a Web-based tool developed by the Center for Applied Special Technology (CAST) at *www.cast.org/bobby*. Bobby checks your page by applying the W3C's Web contents accessibility guidelines to your code and recording the number and type of incompatibility problems it finds. Bobby looks for elements such as consistent use of ALT attributes, appropriate color usage, compatibility with screen readers, and ease of navigation. You can use Bobby online if your pages are live, or you can download Bobby to test your work on your own machine. Unfortunately, many mainstream Web sites fail

Bobby's requirements for accessibility because they use tables as a page layout device and lack support for Cascading Style Sheets. The reluctance of Web designers to adhere to the W3C's accessibility guidelines is directly related to the incompatibility of HTML support across different Web browsers.

You can adopt a browser strategy that enhances your development process. For example, you can code to the last release of HTML when inaugurating a site, and then slowly add enhancements that move you towards the latest release, testing as you go. This method ensures that you are constructing pages that degrade gracefully, that is, pages that users can view in both older and newer browsers. If you want to include animations or effects that require a plug-in, use a development tool that already is supported by the major browsers. Make sure that the most important content on your site is rendered in a way that does not rely on the new technology, so users with older browsers still get your message. Another alternative is to run a browser sniffer, which is a program on the server that detects the user's browser type when they access your site. You then can construct mirrored sets of pages (although this is twice as much work) that are specific to the user's browser. Finally, if you are designing for an intranet, you can work with one browser in mind if you have the luxury of mandating the type of software your readers use.

SHOULD YOU USE AN HTML EDITOR?

You can create or generate HTML code to build Web pages in many ways. In the short history of the Web, the tool that has gained the greatest universal acceptance is Notepad, the simple text editor that comes with Windows 3.1, 95, and 98. On the Macintosh, the equivalent tool is Teach Text or Simple Text. Many sites on the Web are coded using these simple text editing tools. They are easy to use and still relied upon by top-notch HTML authors. They also are the best way to learn HTML because you have to enter every tag by hand.

You also can use a number of HTML editing programs, such as Adobe® PageMill®, NetObjects® Fusion, Microsoft® FrontPage®, and Macromedia Dreamweaver, to name a few. Some code-based HTML editors, such as Allaire HomeSite and Hot Dog Pro, forgo a WYSIWYG approach. They have become popular because they include many powerful enhancements that Notepad lacks, such as multiple search and replace features and syntax checking, while still allowing you to manipulate code at the tag level.

Many of the latest office applications now convert documents to HTML. For example, you can create a flyer in your word processor and export it to create an HTML page. You can even create slides in Microsoft PowerPoint® or Lotus® Freelance Graphics and export them to HTML. This hands-off approach leaves much to be desired for an HTML author because you give up control over the finished product. This type of HTML conversion also is notorious for creating less than standard HTML code. You are better off moving away from one of the office applications to a dedicated HTML authoring package if you are serious about creating attractive, portable Web sites.

As with the browsers, authoring packages interpret tags based on their own built-in logic. Therefore, a page that you create in an editing package may look quite different in the editing interface than it does in a browser. Furthermore, many editing packages create complex, less-than-standard code to achieve an effect specified by the user. The more complex code can cause compatibility problems across different browsers. Remember that HTML is a relatively simple language that is not meant to express complex layouts. Many Web page designers, spoiled by the ease of use of today's powerful word processors, build complex pages with complex text effects and spacing. When the editing program has to translate this for display with simple HTML, it must resort to a variety of methods to accomplish the task. These methods may result in code that is difficult to update or debug. HTML authors who are used to coding by hand (in Notepad or another text editor) often are surprised to see what an HTML editing package has generated for code. To really code effectively with HTML, you must be comfortable working directly at the code level. You may choose to use one of the many editing packages to generate the basic layout or structure for your page or to build a complex table, but be prepared to work with the code and edit at the tag level to fix any discrepancies. You probably will end up working with a combination of tools to create your finished pages.

When in doubt, most experienced HTML authors use Notepad to make final corrections or to quickly fine-tune a section of code. Although the latest HTML editing programs offer many benefits and time-saving features, you cannot rely on them to produce the final look of the page, due to the inconsistent support of HTML in the browsers. When in doubt—test, test, test!

CODING FOR MULTIPLE SCREEN RESOLUTIONS

No matter how carefully you design pages, you can never know how users view your work because you do not know their monitors' screen resolution. A computer monitor's **screen resolution** is the horizontal and vertical height and width of the computer screen in pixels. Most monitors can be set to at least two resolutions, whereas larger monitors have a larger range from which to choose. User screen resolution is a factor over which you have no control.

A monitor's range of screen resolution is a function of the monitor's capabilities and the computer's video card. The three most common screen resolutions (traditionally expressed as width x height) are 640 x 480, 800 x 600, and 1024 x 768. Some users choose to use the highest resolution of 1024 x 768, allowing them to display more on the screen. They may have multiple application windows open at the same time. Users at 800 x 600 may maximize their browser to full screen, whereas those working at 640 x 480 may see additional scroll bars if content does not fit on their screen.

FIXED RESOLUTION DESIGN

Figures 1-2 through 1-4 show the same Web page viewed at different screen resolutions.

FIGURE 1-2
Fixed design at 640 x 480

FIGURE 1-3
Fixed design at 800 x 600

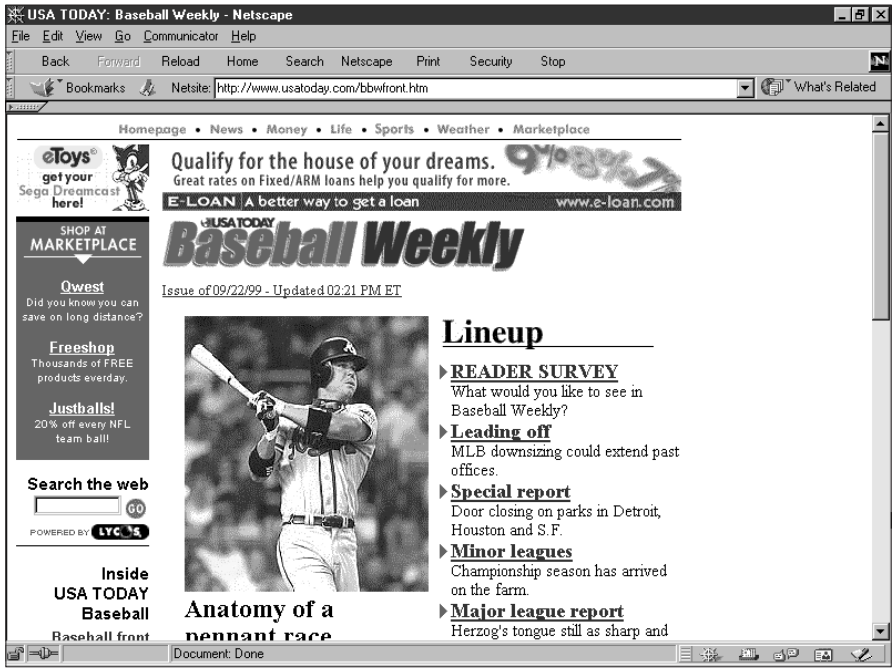


FIGURE 1-4
Fixed design at 1024 x 768

Notice that the page is designed to display its content within 640 x 480 without making the user scroll, indicating that 640 x 480 is the base screen resolution of this Web site. As the screen resolution changes, the content remains aligned to the left side of the page. The negative background white space on the right side of the page fills in the remainder of the screen.

FLEXIBLE RESOLUTION DESIGN

In contrast, Figures 1-5 through 1-7 show a Web page that has been designed to adapt to different screen resolutions.

FIGURE 1-5
Flexible design at 640 x 480

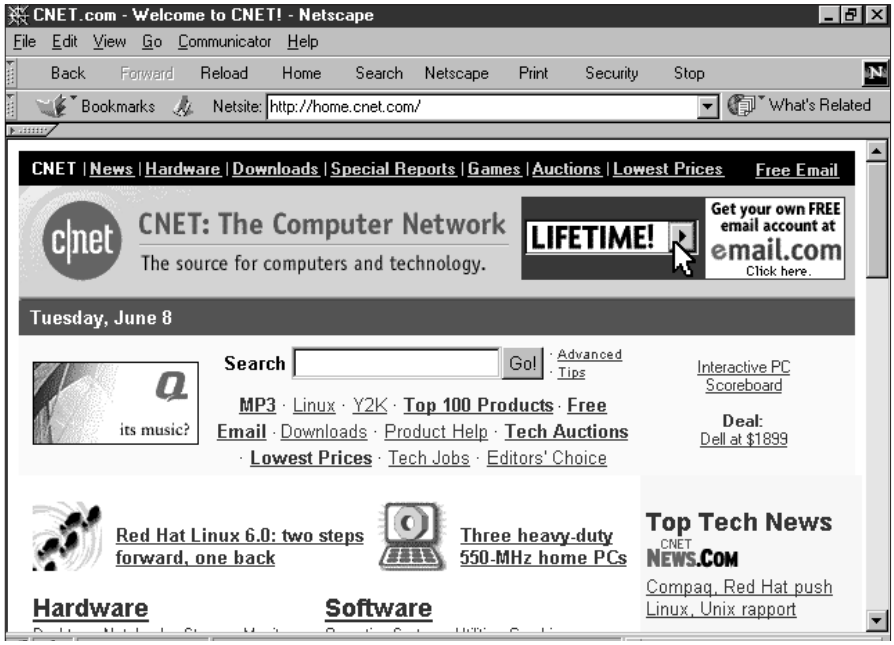
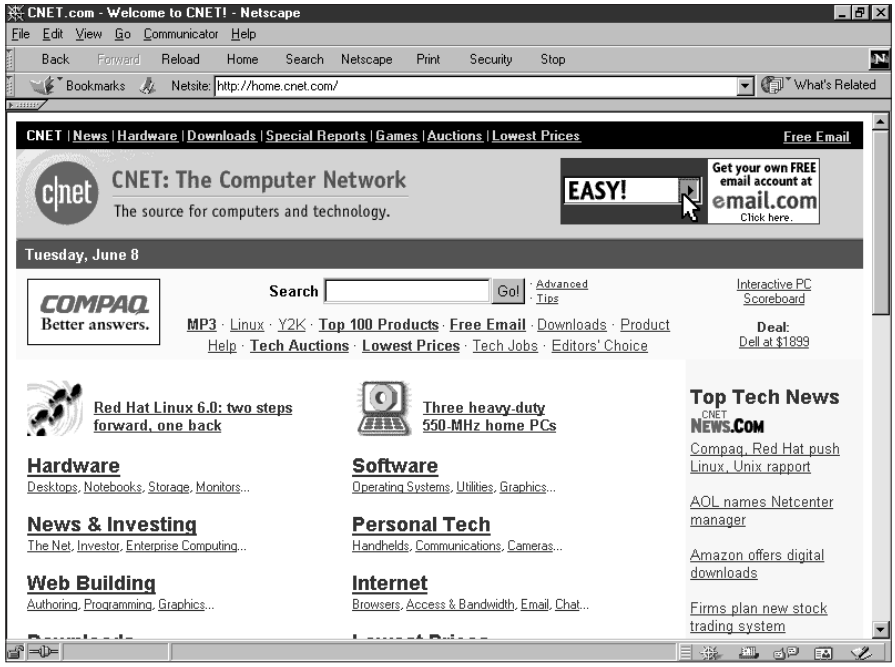


FIGURE 1-6
Flexible design at 800 x 600



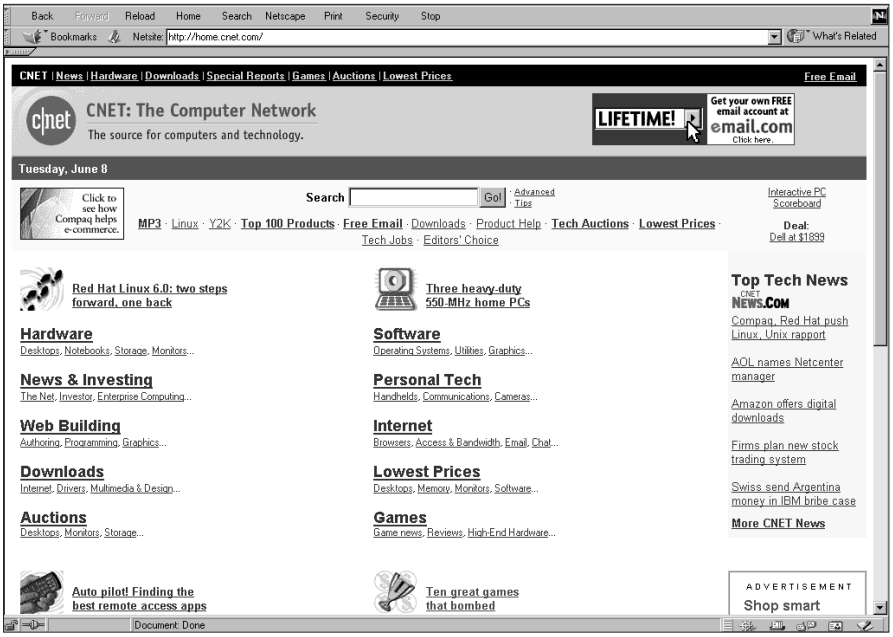


FIGURE 1-7
Flexible design at 1024 x 768

As the screen resolution changes, the white space between the columns expands to accommodate the varying screen width. This space between the columns is more active than the white space in the previous example. The designers accomplished this adaptable solution with variable rather than absolute table widths. You will learn about this technique in Chapter 5.

As a Web designer, decide how you will code your Web site to handle different screen resolutions. Some designers believe, and most major Web sites concur, that you always should code your page to the lowest common denominator, which is currently 640 x 480. Most monitors are delivered set to 640 x 480 resolution. Novice users may not know that they can change their screen resolution. Most experienced computer users find 640 x 480 uncomfortably large and prefer to work at 800 x 600 or better.

Be careful when making the decision to code at higher resolutions. Any content that does not appear with the 640 x 480 window will display additional scroll bars. Although vertical scroll bars are the norm, users consider horizontal scroll bars annoying. Figure 1-8 shows a Web page coded at 800 x 600, and viewed at 640 x 480, with horizontal scroll bars.

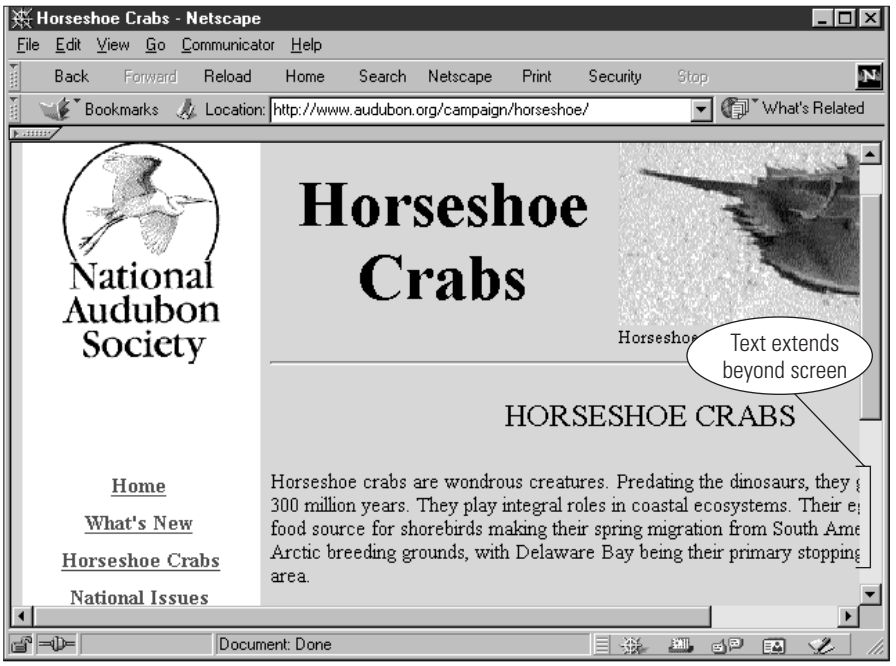


FIGURE 1-8
Page designed at 800 x 600, viewed at 640 x 480

The user must repeatedly scroll from left to right when reading. With more people buying larger monitors, some designers say that 800 x 600 is common enough to use as a base resolution. If you know your audience is consistently using a higher resolution, you can code to it. Otherwise, code to the lowest resolution to make sure that your content displays properly. As always, to ensure that your user can view your pages properly, test at different resolutions.

BANDWIDTH CONCERNS

Although many cable and media providers would have you believe otherwise, it is a myth that most computer users will soon have fast access to the Web. Access via cable modem is currently the most reliable high-speed connection to the Web for home users, but less than 20 percent of American households have access to cable modems. Corporations rely primarily on T1 or Integrated Services Digital Network (ISDN) connections. Digital Subscriber Line (DSL), a new technology that allows voice and high speed Internet access on the same line, is available to only 5 to10 percent of all the households in the U.S. Table 1-1 describes the more common types of connection technologies.

TABLE 1-1
Common types of connection technologies

Technology	Speed	Notes
Regular telephone line	Up to 56 Kbps	This is still the most common method of connecting to the Internet. Most people, however, cannot maintain a connection speed over 44 Kbps.
ISDN basic	64 Kbps to 128 Kbps	ISDN offers good speed, but is fairly expensive. ISDN is more common in urban areas, and primarily used by business.
Digital Subscriber Line (DSL)	512 Kbps to 8 Mbps	For DSL to work well, you must be located within 18,000 feet of your local telephone switching office. DSL uses a single existing phone line to carry both voice and data.
Cable modem	512 Kbps to 52 Mbps	Cable modems are high speed, and allow you to stay connected to the Internet. Because you are not using a phone line, you do not have to dial up to connect. Many cable systems are rushing to install Internet access, but most households will have to wait at least 2 to 5 years.

Connection speed influences your Web page design. Most users simply will not wait longer than 20 seconds for a page to load. If your pages download slowly, your users probably will click to go to another site before they see even a portion of your content. A common mistake that many designers make is to omit testing their pages at different connection speeds. If you do not test, you cannot appreciate what it is like for users to connect at different speeds to your site, and you may lose valuable visitors.

The single biggest factor influencing the speed at which your pages display is the size and number of graphics on your Web pages. Keep your page designs simple with fewer graphics. As a rule of thumb, no single image on your Web site should exceed 10 to 15k. If you know your users all have faster access, you can design your pages to match. For the general public you can consider 28.8 Kbps as a base connection speed because many users still use older modems. You will learn more about how to prepare your images to download quickly in Chapter 7.

HOW PAGES ARE DELIVERED TO THE USER

All Web pages are stored on computers called Web servers. When you type a Uniform Resource Locator (URL) address in your browser, it connects to the appropriate Web server and requests the file you specified. The server serves up the file so your browser can download it. The first time you visit a site, the

entire contents of the HTML file (which is plain text) and every image referenced in the HTML code is downloaded to your hard drive. The next time you visit this site, your browser downloads and parses the HTML file. The browser checks to see if it has any of the specified images stored locally on the computer's hard drive in the cache. The **cache** is the browser's temporary storage area for Web pages and images. The browser always tries to load images from the cache rather than downloading them again from the Web.

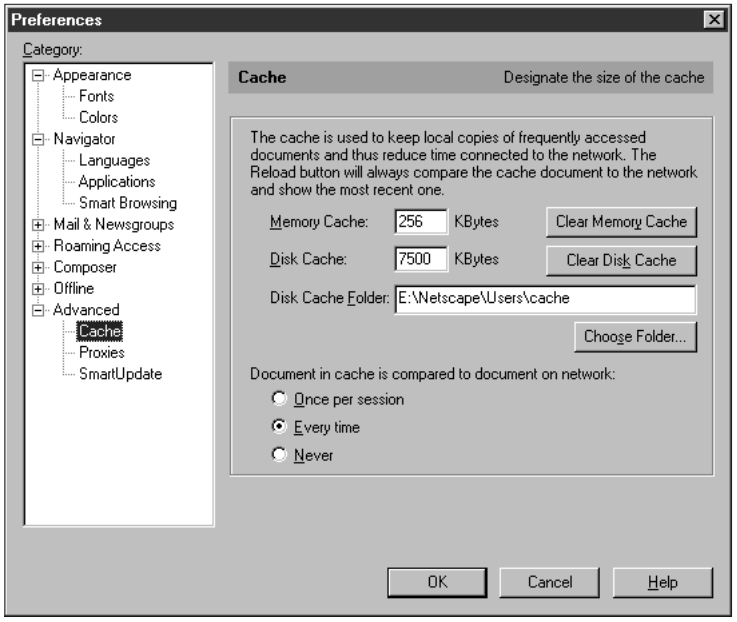
TESTING FOR DOWNLOAD TIMES

Test your site's download time at different connection speeds. Test your site as if you were a user visiting for the first time. This is when users experience the greatest download times. Clear your cache of the files and images that the browser has stored there. The browsers caching mechanism is intended to make the user's browsing experience quicker. As a designer testing your site, however, you want to defeat the natural caching tendencies of the browser. To do this you need to know how to empty your browser's cache.

Emptying the Cache in Netscape 4.x

1. Click **Edit** on the menu bar, and then click **Preferences**.
2. In the Preferences dialog box, click the + next to the Advanced category to expand the category list.
3. In the expanded list, click **Cache** to display the Cache Preferences dialog box, illustrated in Figure 1-9.

FIGURE 1-9
Netscape's Cache Preferences dialog box



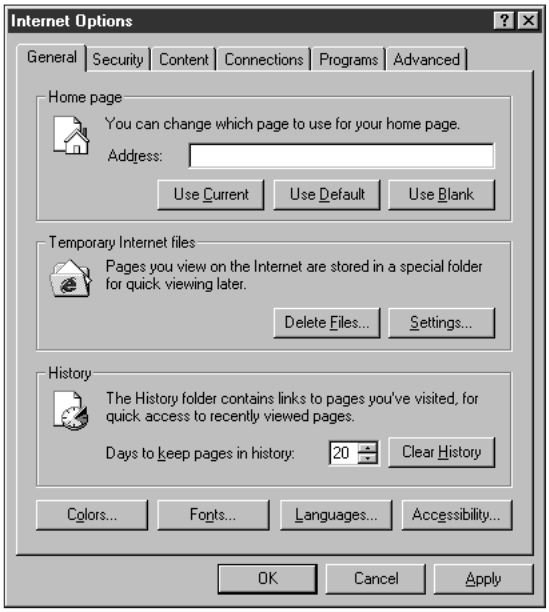
4. In the Cache Preferences dialog box, click the **Clear Memory Cache** button, and then click **OK** to confirm that you want to clear the memory cache.

5. Click the **Clear Disk Cache** button, and then click **OK** to confirm that you want to clear the disk cache.
6. Click **OK** to close the Preferences dialog box.

Emptying the Cache in Internet Explorer 5.0

1. Click **Tools** on the menu bar, and then click **Internet Options**. You see the Internet Options dialog box, which opens to the General tab by default, as illustrated in Figure 1-10.

FIGURE 1-10
Internet Explorer's Internet Options dialog box



2. In the Temporary Internet files section on the General tab, click the **Delete Files** button, and then click **OK**.
3. Click **OK** to close the Internet Options dialog box.

USING THE CACHE

You can use the browser's caching capabilities by reusing graphics as much as possible throughout your site. Once an image is downloaded, it remains in your user's cache for the number of days specified in the user's preference settings. Most users do not change the settings, so there is a good chance your graphics will remain on the user's hard drive a while. Every time the user revisits your site, the cached graphics load locally rather than downloading from the Web server. Leveraging the browser's caching capability is a great argument for standardizing the look of your site by using the same navigation, branding, and background graphics throughout. Not only will the consistency reassure users, your pages will load faster because the browser retrieves the downloaded graphics directly from the cache.

SUMMARY & REVIEW

Many variables affect the way users view your Web pages. As an HTML author, your goal should be to code pages that are accessible to the largest audience possible. As you plan your Web site, make the following decisions before implementing your site.

- Decide which version of HTML you will use to code your pages. Although the latest enhancements to HTML 4.0 offer appealing features, they may not be accessible to all of your users. You may want to code to the HTML 3.2 standard to ensure the greatest portability of your pages. After your site is up and running, you can introduce newer features and functions, testing as you go.
- Decide whether to use Cascading Style Sheets. The style enhancements and control offered by this style language are formidable, but not evenly supported by all browsers. Implement CSS gradually, testing for browser compatibility as you go.
- Choose the suite of browsers you will use to test your site. Although you will include the latest versions of Netscape and Internet Explorer, consider testing in older versions of each browser as well. Test with Opera if your site has a large international audience. If you have a large academic audience, test in Lynx.
- Decide how browser-specific your site will be. Your goal is to create a site that is widely accessible to multiple browsers. If you have a narrow audience or specific requirements, you may want to specify one browser as the primary method for viewing your site.
- Choose the type of editing tool you will use to create your HTML code. You may want to use a WYSIWYG editor to create the general page layout and then rely on Notepad to make corrections to your code. Alternately, a code-based editor such as Hot Dog Pro or Allaire HomeSite lets you work directly with code while enjoying enhancements that Notepad doesn't support.
- Resolve to test your work continually as you build your site. Test with multiple browsers at different screen resolutions and at different connection speeds. If you can, view your site on multiple platforms such as PC, Macintosh, and UNIX as well.

REVIEW QUESTIONS

1. HTML is a subset of which markup language?
2. List three characteristics of HTML that make it ideal for the World Wide Web.
3. What are the benefits of viewing source code on the Web?
4. What work does the World Wide Web Consortium perform?
5. What is a deprecated element?
6. What is a proprietary element?
7. What style language allows the separation of style from structure in HTML?
8. What are the advantages of using an external style sheet?
9. What feature distinguishes XML from HTML?
10. What are the two types of style language designed for use with XML?
11. Explain how XML lends itself to customized data applications.
12. What improvements does XHTML promise over existing HTML?

13. Explain how different browsers affect the display of a Web page.
14. Describe the characteristics of lowest common denominator coding.
15. Describe how coding using the latest technology can prevent users from accessing your site.
16. List the three most common screen resolutions.
17. Explain how screen resolution affects the display of a Web page.
18. List four common types of Internet connection technologies.
19. Explain how the browser's caching capability improves download time.
20. Explain why an HTML author would clear the browser's cache.

PROJECTS

1. Visit the World Wide Web Consortium Web site (*www.w3.org*). Find and describe the three types of HTML 4.0: Transitional, Strict, and Frames, and explain why you might use each.
2. View and copy the source code of a Web site into your text editor. Critique the code for syntax errors and non-standard usage of HTML. Try to determine the version of HTML to which the page is coded.
3. Visit the World Wide Web Consortium Web site (*www.w3.org*). Find the Cascading Style Sheets specification. List and describe ten style properties that you can affect with a style rule.
4. Visit the World Wide Web Consortium site (*www://w3.org*) and examine the Web Accessibility Initiative (WAI). Describe how you would design a page that meets the WAI guidelines.
5. Describe three common mistakes that Web designers make when building a Web site.
6. Test the HTML conversion capabilities of a standard office application.
   a. Use your favorite word processing, spreadsheet, or presentation graphics program that supports conversion to HTML.
   b. Create a document and export it to HTML.
   c. Examine and evaluate the HTML code. Look for non-standard coding techniques or tricks that the program uses to render content into HTML. Write a detailed description of your findings.
7. Test cross-browser compatibility.
   a. Make sure you have recent versions of both Netscape Communicator and Internet Explorer installed on your computer.
   b. Browse a variety of Web sites. Make sure to view various pages of the sites in both browsers.
   c. Write a detailed description of how successfully the various sites appear in both browsers. Look for text, layout, and graphics inconsistencies.
8. Test your browser's caching capabilities.
   a. Visit the front page of a mainstream Web site such as *www.abcnews.com*, *www.nytimes.com*, or *www.wired.com*. Make sure to let the page load completely, allowing the browser to cache all the images on the page. Shut down your browser, then restart it.

b. Visit the page again. Estimate the download time by timing from when you enter the URL until the page has downloaded fully.
c. Clear your browser's cache using the instructions in this chapter. Shut down your browser, and then restart it.
d. Visit the page again. Measure the download time. Note the difference between times for cached and non-cached visits to the site.

CASE STUDY

To complete the ongoing case study for this book, you must create a complete, stand-alone Web site. The site must contain from six to ten pages, displaying at least three levels of information. You can choose your own content. For example, you can do a work-related topic, a personal interest site, or a site for your favorite non-profit organization. The site will be evaluated for cohesiveness, accessibility, and design. At the end of each chapter you will complete a different section of the project. For Chapter 1, get started by creating a project proposal, as outlined below. As you progress through the chapters of the book you will complete different facets of the Web site construction, resulting in a complete Web site.

PROJECT PROPOSAL

Create a one to two page HTML document stating the basic factors or elements you will include in your Web site. Create this document using your favorite HTML editor or Notepad. At this stage your proposal is primarily a draft that you revise. At the end of the next chapter you will have a chance to modify the proposal and supplement the design details.

Include the following items, if applicable:

- Site title — The working title for the site
- Developer — You and anyone else who will work on the site
- Rationale or focus (for example, billboard, customer support, catalog/e-commerce, informational, resource, and so on) — Explain the content and goals of the site. Refer to Chapter 3 for help on content types.
- Main elements outline — List the main features of the site
- Content — Number of individual Web pages
- Target audience — Describe the typical audience for the site
- Design considerations — List the design goals for the site
- Limiting factors — List the technical or audience factors that could limit the design goals of the site